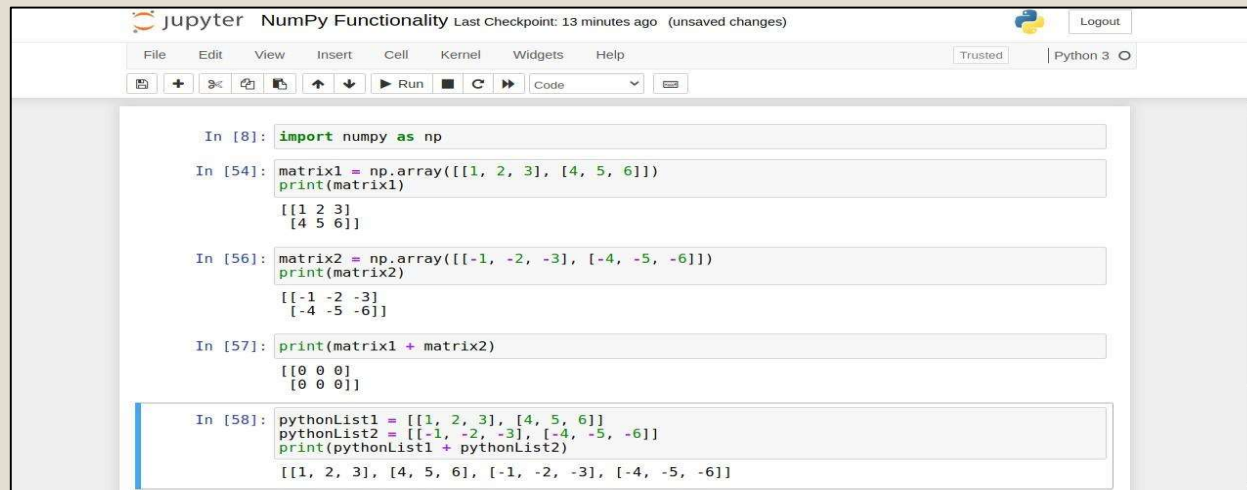


Element-wise Operations

Addition

- We can add the elements of two NumPy arrays simply by using (+) operator.
- Each element of the first array is added to the corresponding element of the second array.
- Note that adding two Python lists using plus (+) operator is not possible. Instead the lists are concatenated if we use (+) operator.



The screenshot shows a Jupyter Notebook interface with the title 'NumPy Functionality'. The notebook contains five code cells. The first cell imports NumPy as 'np'. The second cell creates a 2x2 NumPy array 'matrix1' with values [[1, 2, 3], [4, 5, 6]] and prints it. The third cell creates another 2x2 NumPy array 'matrix2' with values [[-1, -2, -3], [-4, -5, -6]] and prints it. The fourth cell prints the result of adding 'matrix1' and 'matrix2', which is a 2x2 array of zeros. The fifth cell demonstrates that adding two Python lists results in concatenation, showing 'pythonList1 + pythonList2' as '[[1, 2, 3], [4, 5, 6], [-1, -2, -3], [-4, -5, -6]]'.

```
In [8]: import numpy as np

In [54]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
         print(matrix1)
[[1 2 3]
 [4 5 6]]

In [56]: matrix2 = np.array([[-1, -2, -3], [-4, -5, -6]])
         print(matrix2)
[[-1 -2 -3]
 [-4 -5 -6]]

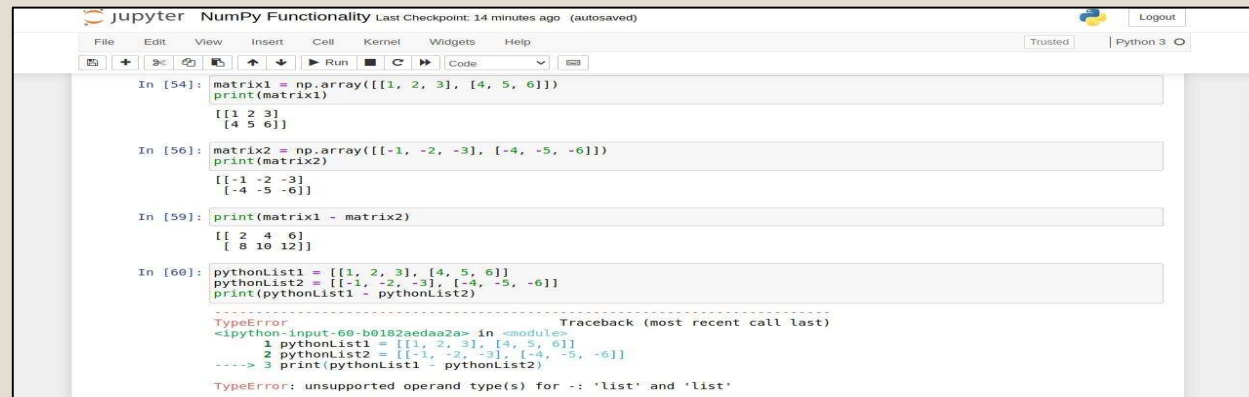
In [57]: print(matrix1 + matrix2)
[[0 0 0]
 [0 0 0]]

In [58]: pythonList1 = [[1, 2, 3], [4, 5, 6]]
         pythonList2 = [[-1, -2, -3], [-4, -5, -6]]
         print(pythonList1 + pythonList2)
[[1, 2, 3], [4, 5, 6], [-1, -2, -3], [-4, -5, -6]]
```

Element-wise Operations

Subtraction

- We can subtract the elements of two NumPy arrays simply by using (-) operator.
- Each element of the second array is subtracted from the corresponding element of the first array.
- Note that subtracting the elements of two Python lists using (-) operator will result in an error.



The screenshot shows a Jupyter Notebook interface with the title 'NumPy Functionality'. It contains four code cells. The first three cells demonstrate NumPy array subtraction, while the fourth cell shows an error when attempting to subtract Python lists.

```
In [54]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix1)
[[1 2 3]
 [4 5 6]]

In [56]: matrix2 = np.array([[1, -2, -3], [-4, -5, -6]])
print(matrix2)
[[1 -2 -3]
 [-4 -5 -6]]

In [59]: print(matrix1 - matrix2)
[[ 2  4  6]
 [ 8 10 12]]

In [60]: pythonList1 = [[1, 2, 3], [4, 5, 6]]
pythonList2 = [[1, -2, -3], [-4, -5, -6]]
print(pythonList1 - pythonList2)

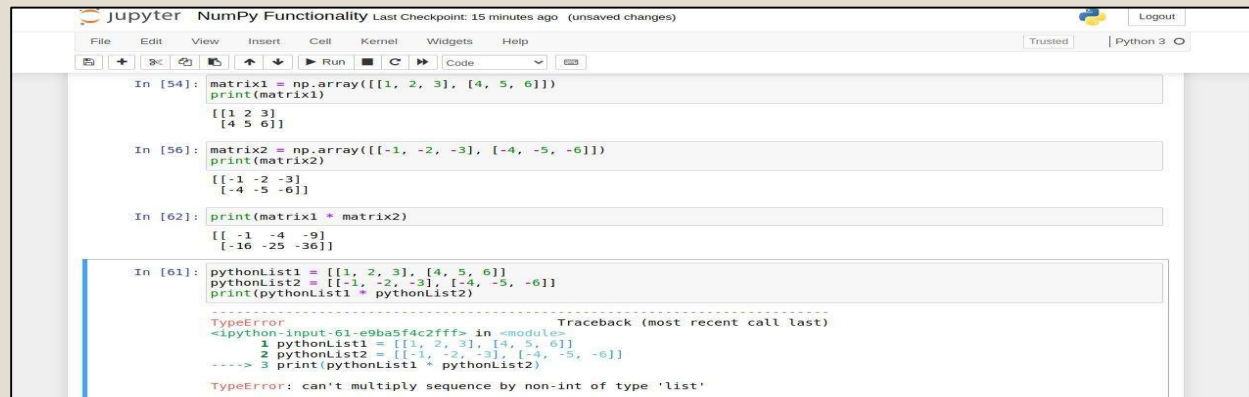
TypeError                                 Traceback (most recent call last)
<ipython-input-60-b0182aedaa2a> in <module>
      1 pythonList1 = [[1, 2, 3], [4, 5, 6]]
      2 pythonList2 = [[1, -2, -3], [-4, -5, -6]]
----> 3 print(pythonList1 - pythonList2)

TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

Element-wise Operations

Multiplication

- We can multiply the elements of two NumPy arrays simply by using (*) operator.
- Each element of the first array is multiplied by the corresponding element of the second array.
- Note that multiplying the elements of two Python lists using (*) operator will result in an error.



The screenshot shows a Jupyter Notebook interface with the title 'NumPy Functionality'. It contains four code cells. The first three cells demonstrate NumPy array multiplication, while the fourth cell shows an error when attempting to multiply Python lists.

```
jupyter NumPy Functionality Last Checkpoint: 15 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
In [54]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
         print(matrix1)
         [[1 2 3]
          [4 5 6]]

In [56]: matrix2 = np.array([[-1, -2, -3], [-4, -5, -6]])
         print(matrix2)
         [[-1 -2 -3]
          [-4 -5 -6]]

In [62]: print(matrix1 * matrix2)
         [[-1 -4 -9]
          [-16 -25 -36]]

In [61]: pythonList1 = [[1, 2, 3], [4, 5, 6]]
         pythonList2 = [[-1, -2, -3], [-4, -5, -6]]
         print(pythonList1 * pythonList2)

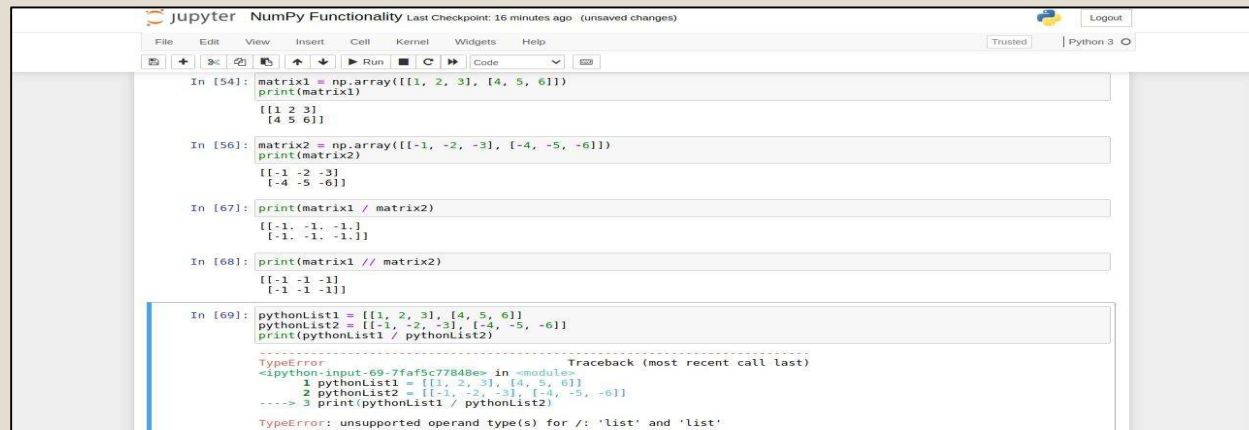
         TypeError                                 Traceback (most recent call last)
         <ipython-input-61-e9ba5f4c2fff> in <module>
             1 pythonList1 = [[1, 2, 3], [4, 5, 6]]
             2 pythonList2 = [[-1, -2, -3], [-4, -5, -6]]
         ----> 3 print(pythonList1 * pythonList2)

         TypeError: can't multiply sequence by non-int of type 'list'
```

Element-wise Operations

Division

- We can divide the elements of two NumPy arrays simply by using (/) operator.
- Each element of the first array is divided by the corresponding element of the second array.
- Note that dividing the elements of two Python lists using (/) operator will result in an error.



The screenshot shows a Jupyter Notebook interface with the title 'jupyter NumPy Functionality'. The notebook contains five code cells. The first four cells demonstrate NumPy array creation and element-wise division. The fifth cell attempts to divide two Python lists, resulting in a `TypeError`.

```
In [54]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrix1)
[[1 2 3]
 [4 5 6]]

In [56]: matrix2 = np.array([[1, -2, -3], [-4, -5, -6]])
print(matrix2)
[[1 -2 -3]
 [-4 -5 -6]]

In [67]: print(matrix1 / matrix2)
[[1. -1. -1.]
 [-1. -1. -1.]]

In [68]: print(matrix1 // matrix2)
[[1 -1 -1]
 [-1 -1 -1]]

In [69]: pythonList1 = [[1, 2, 3], [4, 5, 6]]
pythonList2 = [[1, -2, -3], [-4, -5, -6]]
print(pythonList1 / pythonList2)

TypeError                                 Traceback (most recent call last)
<ipython-input-69-7faf5c77848e> in <module>
      1 pythonList1 = [[1, 2, 3], [4, 5, 6]]
      2 pythonList2 = [[1, -2, -3], [-4, -5, -6]]
----> 3 print(pythonList1 / pythonList2)

TypeError: unsupported operand type(s) for /: 'list' and 'list'
```

Matrix Multiplication

- Apart from elementwise multiplication, NumPy also provides us with a built-in function to compute the matrix multiplication of two arrays.
- We use the `.matmul()` function inside the NumPy library for matrix multiplication of two arrays.

A screenshot of a Jupyter Notebook interface titled "NumPy Functionality". The interface includes a top menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations and a "Run" button. The notebook contains three code cells. The first cell defines a 3x3 matrix 'matrix1' and prints it. The second cell defines a 3x3 matrix 'matrix2' and prints it. The third cell uses 'np.matmul()' to multiply 'matrix1' and 'matrix2', with the result displayed below the code.

```
jupyter NumPy Functionality Last Checkpoint: 19 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [75]: matrix1 = np.array([[1, 2, 3], [4, 5, 6], [0, 0, 0]])
print(matrix1)

[[1 2 3]
 [4 5 6]
 [0 0 0]]

In [76]: matrix2 = np.array([[-1, -2, -3], [-4, -5, -6], [0, 0, 0]])
print(matrix2)

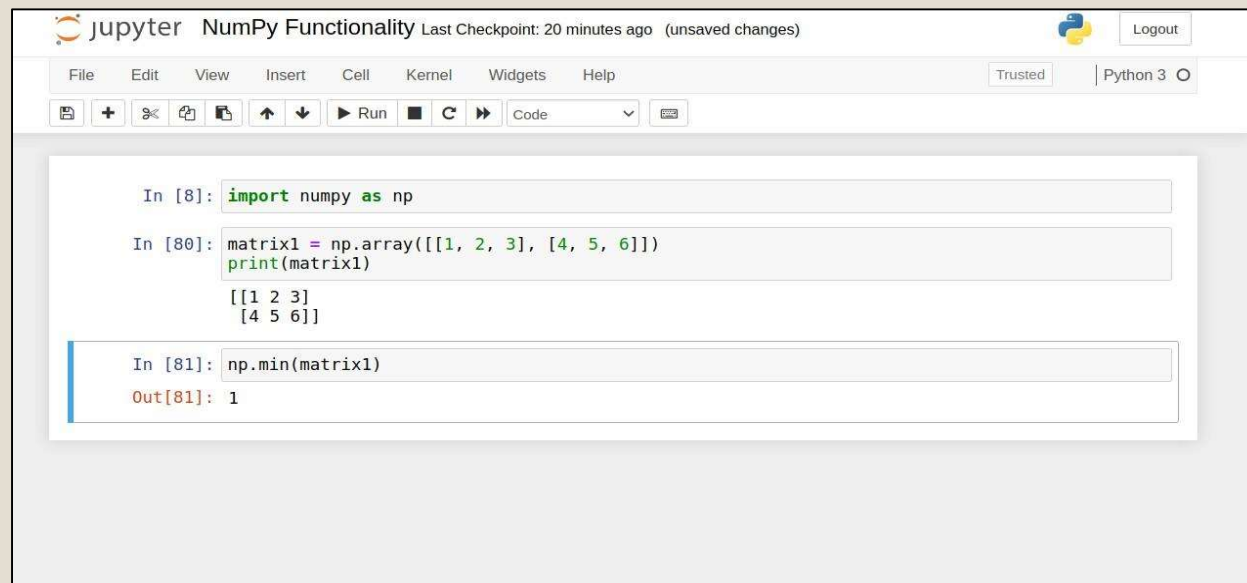
[[-1 -2 -3]
 [-4 -5 -6]
 [ 0  0  0]]

In [77]: np.matmul(matrix1, matrix2)
Out[77]: array([[ -9, -12, -15],
                [-24, -33, -42],
                [  0,   0,   0]])
```

Statistics

.min()

- .min() function gives us the minimum value in a NumPy array.
- This function can also be applied on Python lists.



The image shows a Jupyter Notebook interface with the title "NumPy Functionality". The top bar includes a "Logout" button and a "Trusted" status indicator. The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar has icons for saving, adding cells, undo, redo, and running code. The code area contains three cells:

```
In [8]: import numpy as np

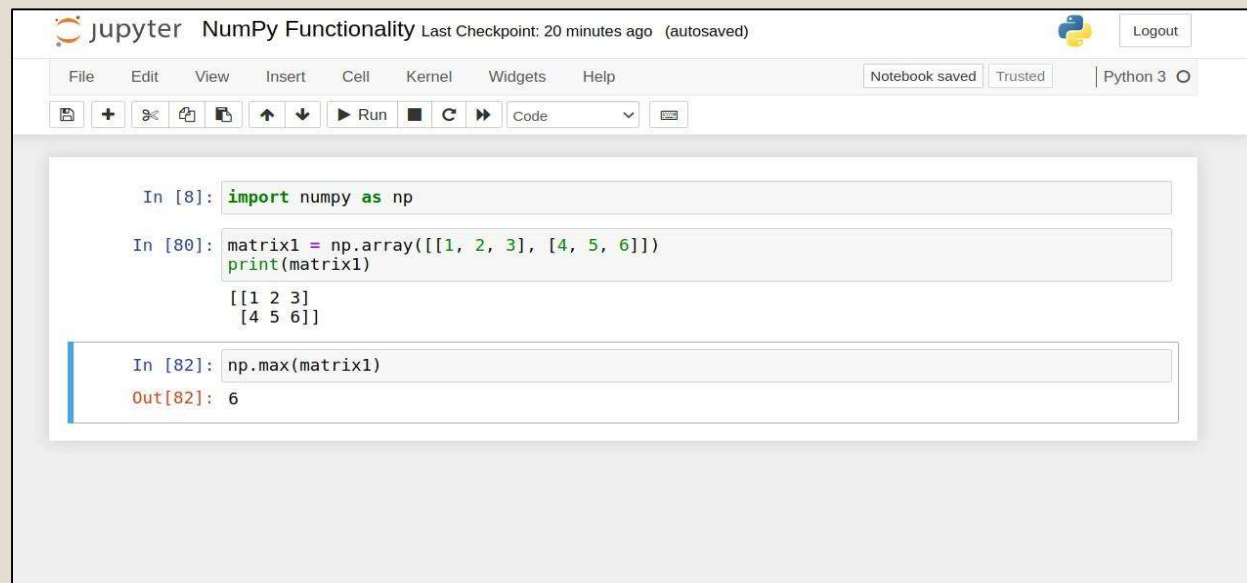
In [80]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
          print(matrix1)
          [[1 2 3]
           [4 5 6]]

In [81]: np.min(matrix1)
Out[81]: 1
```

Statistics

.max()

- .max() function gives us the maximum value in a NumPy array.
- This function can also be applied on Python lists.



The image shows a Jupyter Notebook interface with the title "NumPy Functionality". The notebook contains three code cells. The first cell imports NumPy as np. The second cell creates a 2x3 NumPy array named matrix1 and prints it, showing the output as [[1 2 3] [4 5 6]]. The third cell uses np.max(matrix1) to find the maximum value in the array, and the output is displayed as 6.

```
In [8]: import numpy as np

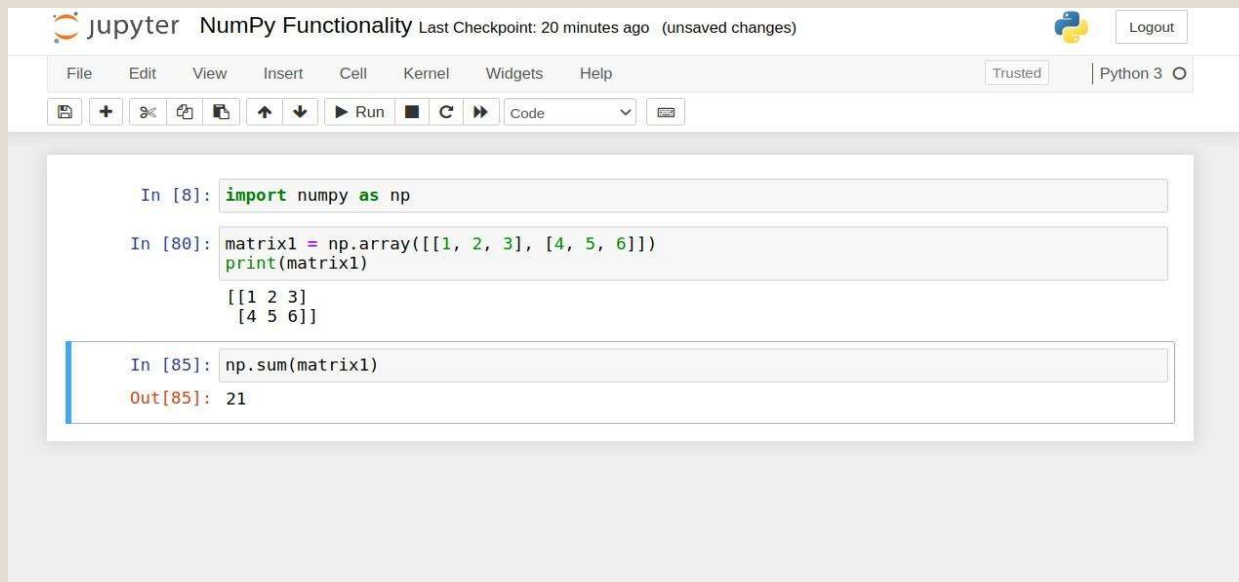
In [80]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
          print(matrix1)
          [[1 2 3]
           [4 5 6]]

In [82]: np.max(matrix1)
Out[82]: 6
```

Statistics

.sum()

- .sum() function gives us the sum of all the values in a NumPy array.
- This function can also be applied on Python lists.

A screenshot of a Jupyter Notebook interface. The top bar shows the Jupyter logo, the title 'NumPy Functionality', and a status message 'Last Checkpoint: 20 minutes ago (unsaved changes)'. On the right, there is a 'Logout' button. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A toolbar with various icons for file operations and execution is located below the menu bar. The main area contains three code cells. The first cell (In [8]) contains 'import numpy as np'. The second cell (In [80]) contains 'matrix1 = np.array([[1, 2, 3], [4, 5, 6]])' followed by 'print(matrix1)', which outputs a 2x3 array. The third cell (In [85]) contains 'np.sum(matrix1)', which outputs '21'.

```
jupyter NumPy Functionality Last Checkpoint: 20 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [8]: import numpy as np

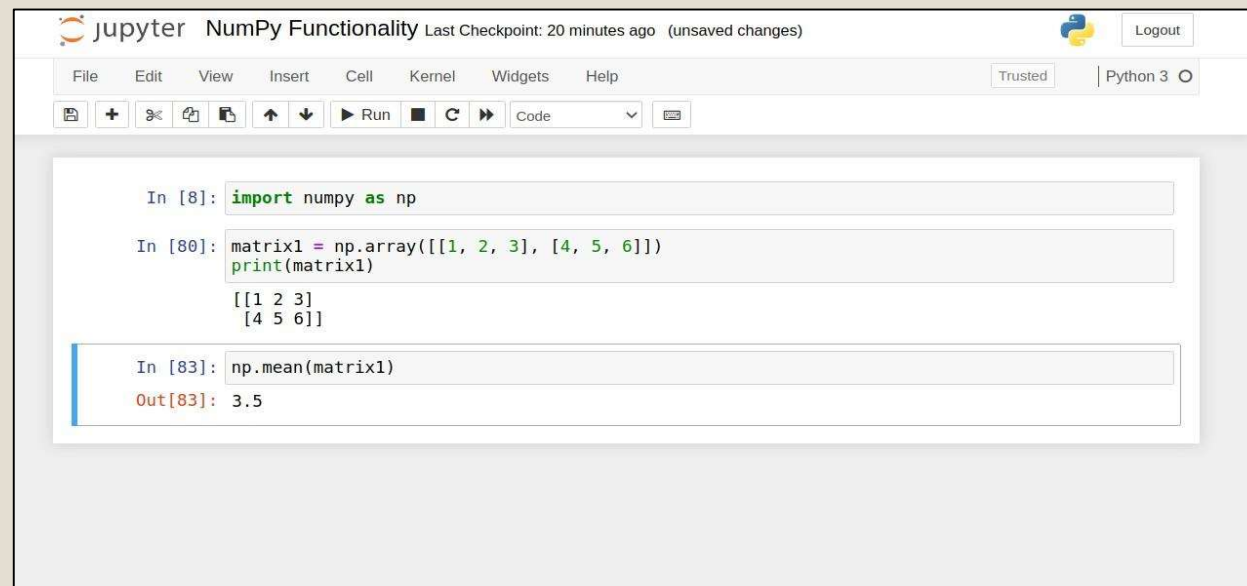
In [80]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
          print(matrix1)
          [[1 2 3]
           [4 5 6]]

In [85]: np.sum(matrix1)
Out[85]: 21
```


Statistics

`.mean()`

- `.mean()` function gives us the mean of all the values in a NumPy array.
- This function can also be applied on Python lists.



A screenshot of a Jupyter Notebook interface titled "NumPy Functionality". The notebook shows three code cells. The first cell imports NumPy as `np`. The second cell creates a 2x3 NumPy array named `matrix1` with values `[[1, 2, 3], [4, 5, 6]]` and prints it. The third cell calculates the mean of `matrix1` using `np.mean(matrix1)`, resulting in the output `3.5`. The interface includes a top menu bar with options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below the menu contains icons for file operations, running, and other notebook functions. The status bar at the bottom indicates the kernel is "Python 3" and shows a "Logout" button.

```
In [8]: import numpy as np

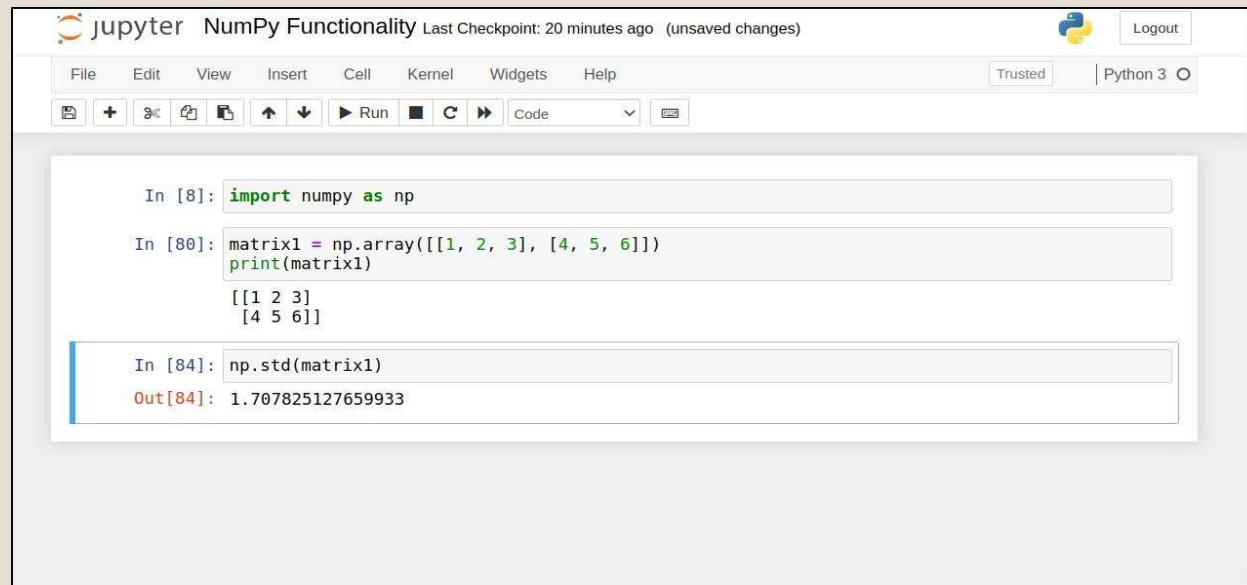
In [80]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
          print(matrix1)
          [[1 2 3]
           [4 5 6]]

In [83]: np.mean(matrix1)
Out[83]: 3.5
```

Statistics

.std()

- .std() function gives us the standard deviation of a NumPy array.
- This function can also be applied on Python lists.



```
jupyter NumPy Functionality Last Checkpoint: 20 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [8]: import numpy as np

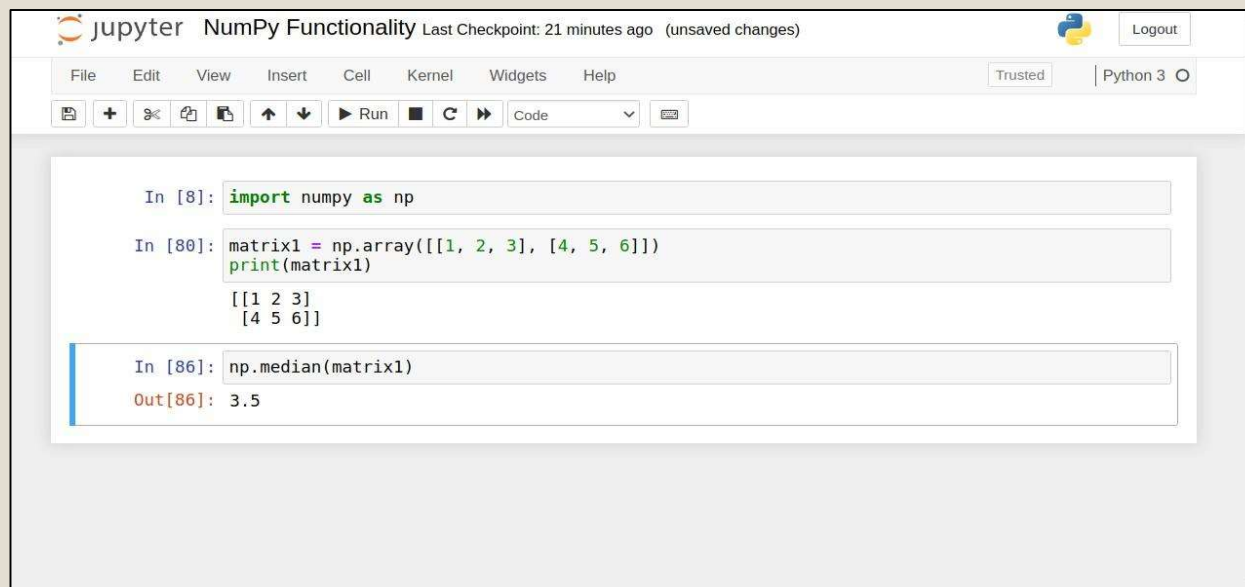
In [80]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
          print(matrix1)
          [[1 2 3]
           [4 5 6]]

In [84]: np.std(matrix1)
Out[84]: 1.707825127659933
```

Statistics

`.median()`

- `.median()` function gives us the median of a NumPy array.
- This function can also be applied on Python lists.

A screenshot of a Jupyter Notebook interface. The title bar shows 'jupyter NumPy Functionality' and 'Last Checkpoint: 21 minutes ago (unsaved changes)'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The toolbar contains icons for file operations, running, and code execution. The notebook content shows three input cells. The first cell contains 'import numpy as np'. The second cell contains 'matrix1 = np.array([[1, 2, 3], [4, 5, 6]])' followed by 'print(matrix1)', which outputs a 2x3 array. The third cell contains 'np.median(matrix1)', which outputs '3.5'.

```
In [8]: import numpy as np

In [80]: matrix1 = np.array([[1, 2, 3], [4, 5, 6]])
          print(matrix1)
          [[1 2 3]
           [4 5 6]]

In [86]: np.median(matrix1)
Out[86]: 3.5
```